Springboard DSC Program

Capstone Project 1

Credit Card Fraud Detection

Anonymized credit card transactions labeled as fraudulent or genuine

Credit Card Fraud Detection Problem

By Harinee Madhusudhan

May2020

Table of Contents

Introduction

The objective of the Credit Card Fraud Detection Problem is to use available data to predict if a future card transaction is valid or a fraudulent one. The prediction is based on an artificial intelligence model using past credit card transactions that are labeled with information if the transaction is a fraudulent one or not.

The dataset at https://www.kaggle.com/mlg-ulb/creditcardfraud was used to model a machine language application.

The implementation details can be found in the notebooks I developed, which at at <GIT Location>

Approach

Data Acquisition and Wrangling

The data set has a total of 284807 records of which 492 are fraud and the remaining 284315 are valid transactions. In addition, the data set is already normalized, with 28 variables V1 to V28, scaled to a mean of 0. The other variables are Time and Amount which are numeric values. All missing values were already filled in, so there are no gaps in any of the data items.

The credit card transactions are for two's consecutive days. The time column contains the seconds from the start of data collection. More transactions occur during the time between 9:00 AM to 11:00 PM on both days compared to the other times.

Looking at the distribution of Amounts, a vast majority of transactions were very low. Only about 1% of the transactions were above $1,000 and 3% of the transactions were above $500.

This distribution is expected since daily transactions aren't extremely expensive, but that's where most fraudulent transactions happen.

There are only 492 fraudulent transactions out of 28.4K transactions – which is about 0.173% of all of the transactions. The incidence of fraud is very less, so the data is highly imbalanced.

```
No Frauds 99.83 % of the dataset Frauds 0.17 % of the dataset
```

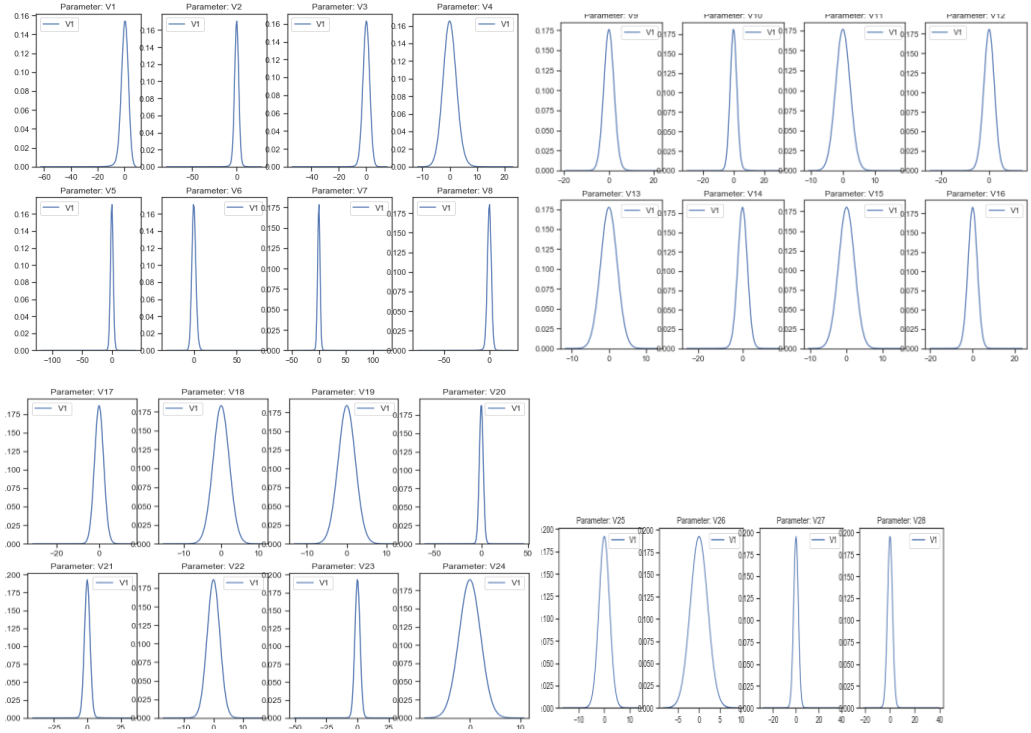
Storytelling and Inferential Statistics

From the data available, it looks like the fraudulent transactions occur continuously, interspersed between the good transactions.

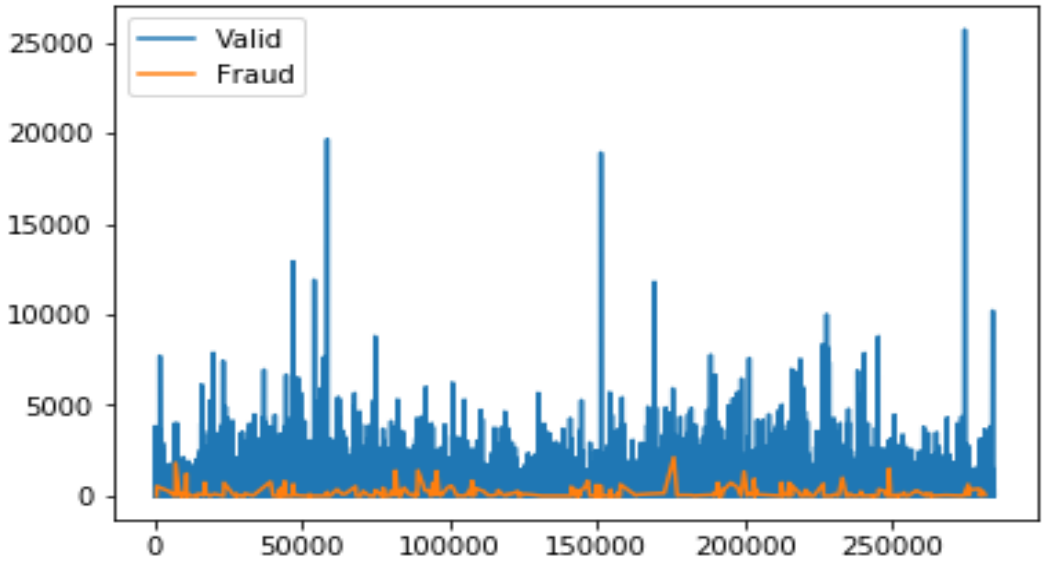
Which of the 28 variables have more impact in the determination of fraudulent transactions?

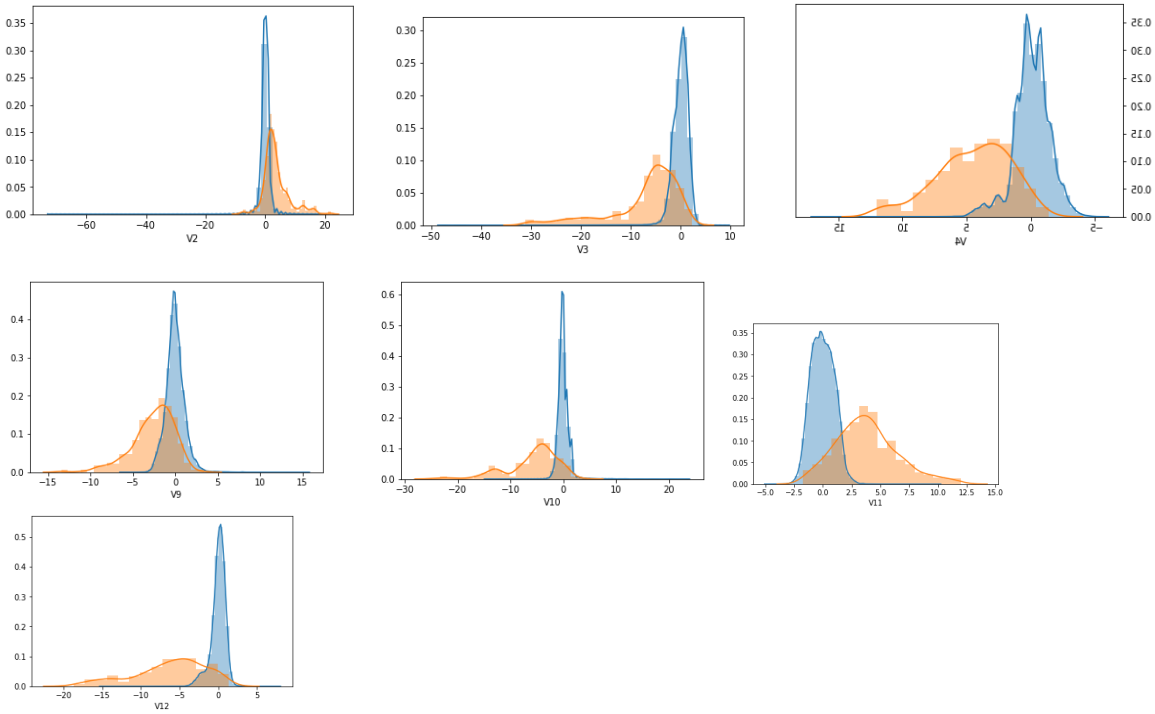
To respond to this question, we draw the histogram distributions for good and fraudulent transactions with a specific variable and see if the distributions overlap or are distinct. The more distinct they are, the more effect the variable has on the determination of fraudulent transactions.

How are these parameters distributed? They all seem to be normally distributed, with a skew towards the right.



Looks like fraudulent transactions are of lower denominations only. Is there a specific time period when fraudulent transactions occur.





Baseline Modeling

The first step in the modeling process was to split the data into training and test set. Used the following command to split the data with 80% of them in the training dataset and the rest in the test dataset.

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=123, shuffle=True)

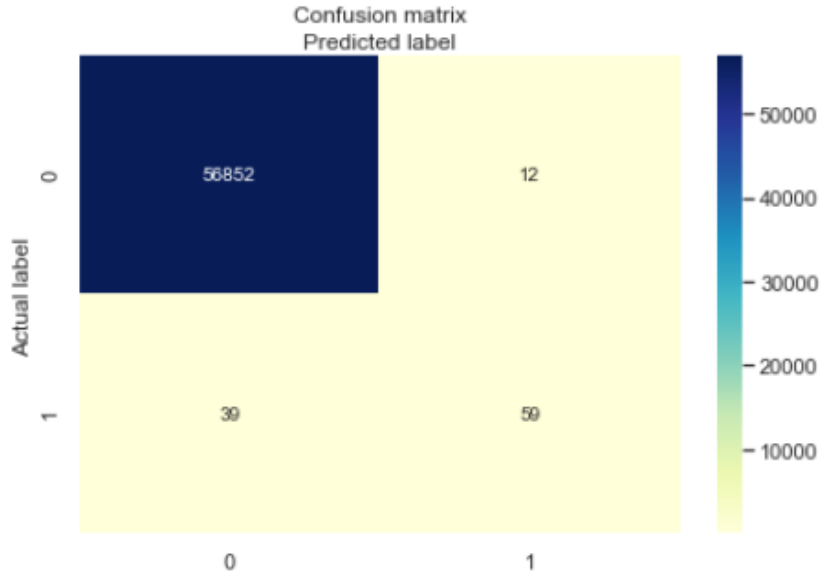
The resultant split was as below:

- Training Data
    - No Frauds 99.83 % of the dataset -- 227451 of 227845
    - Frauds 0.17 % of the dataset -- 394 of 227845
- Test Data
    - No Frauds 99.83 % of the dataset -- 56864 of 56962
    - Frauds 0.17 % of the dataset -- 98 of 56962

The next step was to run a baseline logistic regression. Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression).

I ran a simple Logistic Regression with 500 iterations, and fitted the training data to the model. The confusion matrix was as below:

```
[56852  12]
[ 39  59]
```



The ROC and the training classification are as below:

```
[Training Classification Report]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    227451
           1       0.85      0.64      0.73       394

    accuracy                           1.00    227845
   macro avg       0.92      0.82      0.86    227845
weighted avg       1.00      1.00      1.00    227845

[Test Classification Report]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.83      0.60      0.70        98

    accuracy                           1.00     56962
   macro avg       0.92      0.80      0.85     56962
weighted avg       1.00      1.00      1.00     56962
```

The ROC looks reasonably good for an initial model. The area under ROC was 0.91 – and we could try improving this.

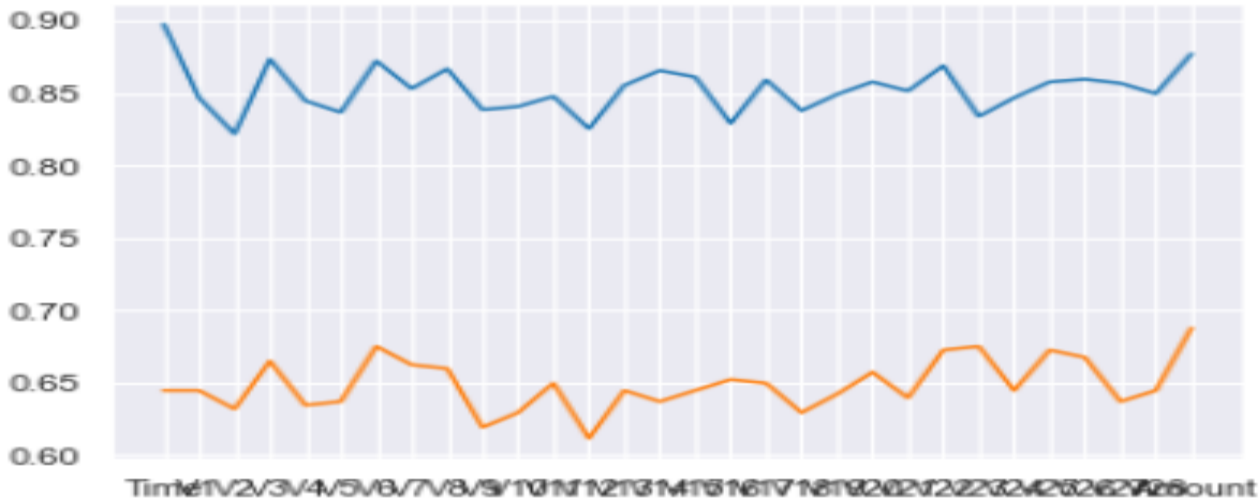
The training and test precision/recall for the fraud situation was 0.85/0.64 and 0.83/0.60. These are low. However, for the training dataset, the precision/recall are close to 1. This is because of the large number of valid transactions in the dataset.

Extended Modeling

In order to make the model better, I need to perform two tasks:

1. Drop any unnecessary variables in the dataset and retry the modeling
2. Handle the imbalance in the data

Dropping Variables:
I recreated the training and test data sets, by dropping one of the variables and tried the same model as above. Here is a chart of the precision and recall.



It is seen that there is some improvement in the precision/recall, when some variable are dropped out, however nothing very distinctive.

This imbalance and bias in the training dataset influences the algorithms, leading to reduced. This is a problem as it is typically the minority class on which predictions are most important.

One approach to addressing the problem of class imbalance is to randomly resample the training dataset. I will follow two main approaches to randomly resample the imbalanced dataset. The first one is undersampling, by deleting rows from the valid transactions. The second one, oversampling, would be to add rows to fraudulent transactions by duplicating existing fraudulent ones. In both cases, the objective would be to equal the number of valid and fraudulent transactions in the training dataset.

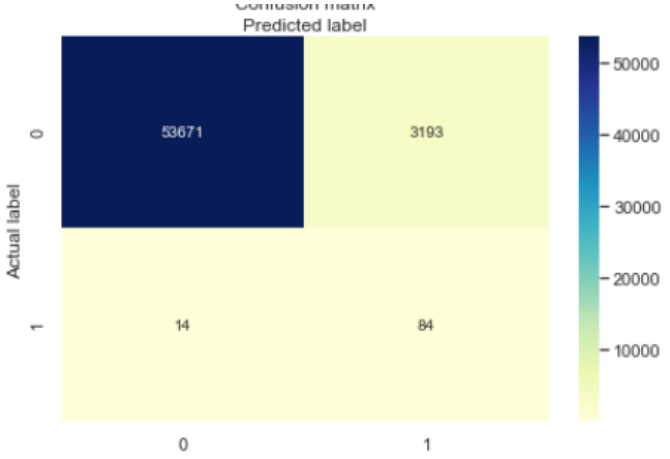
Random Undersampling:

I undersampled the valid transaction training data set, so the training dataset looks as:

```
788, 30
```

I applied the same model as above to get the following confusion matrix and ROC – using the undersampled training dataset.

```
[53671 3193]
[ 14  84]
```



The area under ROC has improved to 0.95. The classification report is as below:

```
Training Classification Report
              precision    recall  f1-score   support

           0       0.91      0.96      0.93       394
           1       0.95      0.90      0.93       394

    accuracy                           0.93       788
   macro avg       0.93      0.93      0.93       788
weighted avg       0.93      0.93      0.93       788

Test Classification Report
              precision    recall  f1-score   support

           0       1.00      0.94      0.97     56864
           1       0.03      0.86      0.05        98

    accuracy                           0.94     56962
   macro avg       0.51      0.90      0.51     56962
weighted avg       1.00      0.94      0.97     56962
```

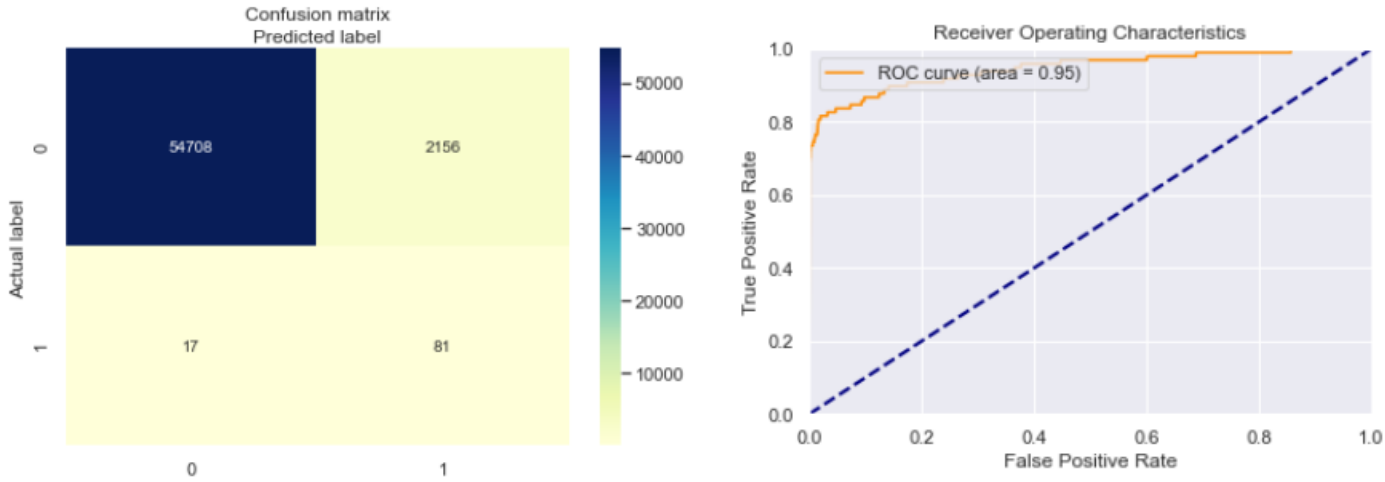
Random Oversampling:

I oversampled the valid transaction training data set, so the training dataset looks as:

```
(454902, 30)
```

I applied the same model as above to get the following confusion matrix and ROC – using the oversampled training dataset.

```
[[54708 2156] [ 17 81]]
```



The area under ROC is 0.96. The classification report is as below:

```
Training Classification Report
              precision    recall  f1-score   support

           0       0.90      0.96      0.93    227451
           1       0.96      0.89      0.92    227451

    accuracy                           0.93    454902
   macro avg       0.93      0.93      0.93    454902
weighted avg       0.93      0.93      0.93    454902

Test Classification Report
              precision    recall  f1-score   support

           0       1.00      0.96      0.98     56864
           1       0.04      0.83      0.07        98

    accuracy                           0.96     56962
   macro avg       0.52      0.89      0.52     56962
weighted avg       1.00      0.96      0.98     56962
```

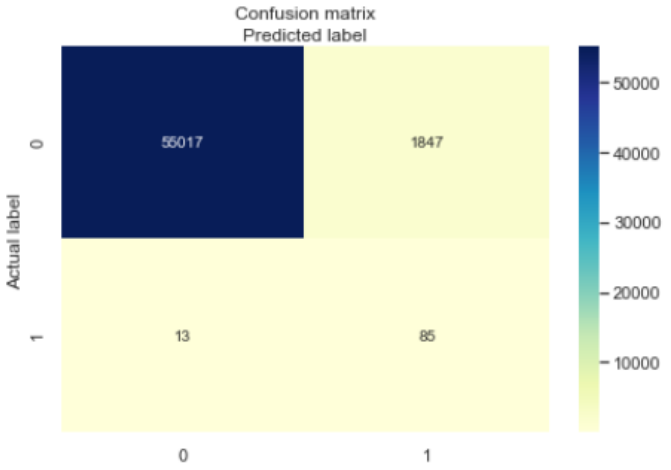
Imbalance Learn

Using the imbalanced-learn package, a better set of resampled data could be obtained for such imbalanced sets. Using the RandomUnderSampler and RandomOverSampler modules from the imbalanced-learn package, I resampled two further sets of training data.

Undersampling:

When I ran the same model with undersampled data using imblearn, the confusion matric was as below:

```
[55017 1847]
[ 13  85]
```



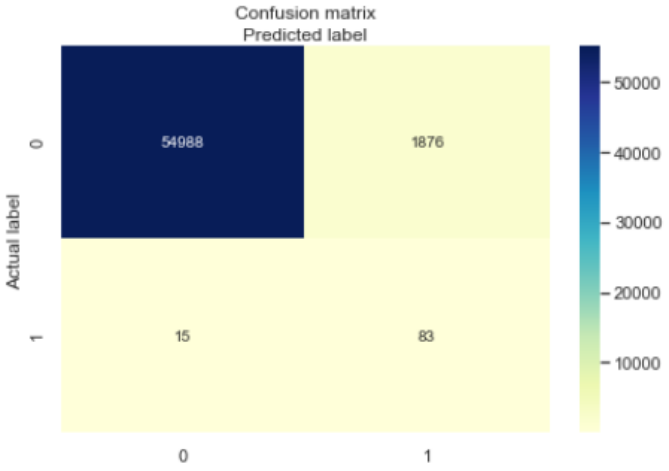
The area under ROC is 0.97. The classification report is as below:

```
  Training Classification Report
               precision    recall  f1-score   support

            0       0.93      0.97      0.95       394
            1       0.97      0.93      0.95       394

     accuracy                           0.95       788
    macro avg       0.95      0.95      0.95       788
 weighted avg       0.95      0.95      0.95       788

  Test Classification Report
               precision    recall  f1-score   support

            0       1.00      0.97      0.98     56864
            1       0.04      0.87      0.08        98

     accuracy                           0.97     56962
    macro avg       0.52      0.92      0.53     56962
 weighted avg       1.00      0.97      0.98     56962
```

Oversampling

When I ran the same model with oversampled data using imblearn, the confusion matric was as below

```
[54988 1876]

[ 15 83]
```



The area under ROC is 0.96. The classification report is as below:

```
Training Classification Report
              precision    recall  f1-score   support

           0       0.93      0.97      0.95    227451
           1       0.96      0.93      0.95    227451

    accuracy                           0.95    454902
   macro avg       0.95      0.95      0.95    454902
weighted avg       0.95      0.95      0.95    454902

Test Classification Report
              precision    recall  f1-score   support

           0       1.00      0.97      0.98     56864
           1       0.04      0.85      0.08        98

    accuracy                           0.97     56962
   macro avg       0.52      0.91      0.53     56962
weighted avg       1.00      0.97      0.98     56962
```

Findings

Here is the list of training metrics:

Model	Class	precision	recall	f1-score	support
Initial Run	0	0.999380345	0.99979776	0.999589008	227451
Initial Run	1	0.846153846	0.64213198	0.73015873	394
Random Undersampling	0	0.908433735	0.956852792	0.932014833	394
Random Undersampling	1	0.954423592	0.903553299	0.928292047	394
Random Oversampling	0	0.8960038	0.961912676	0.927789192	227451
Random Oversampling	1	0.958888573	0.888353975	0.922274637	227451
IMBLearn Undersampling	0	0.929440389	**0.969543147**	0.949068323	394
IMBLearn Undersampling	1	0.968169761	**0.926395939**	0.946822309	394
IMBLearn Oversampling	0	0.931274233	**0.**965188106	0.947927934	227451
IMBLearn Oversampling	1	0.963872463	0.928771472	0.945996476	227451

Looking at the above tables, I used different sets of data items for training the model. In case of the initial run, I used a training data set with 227451 rows of valid transactions and 394 rows of fraudulent transactions.

For Random undersampling and IMBLearn undersampling, I used a training data set with 394 rows of valid and fraudulent transactions

For Random Oversampling and IMBLearn oversampling, I used a training data set with 227451 rows of valid and fraudulent transactions

Here is the list of test metrics:

Model	**Class**	**precision**	**recall**	**f1-score**	**support**
Initial Run	0	0.999314479	0.99978897	0.999551668	56864
Initial Run	1	0.830985915	0.602040816	0.698224852	98
Random Undersampling	0	0.99973922	0.943848481	0.97099024	56864
Random Undersampling	1	0.025633201	0.857142857	0.049777778	98
Random Oversampling	0	0.999689356	0.962084975	0.980526754	56864
Random Oversampling	1	0.036209209	0.826530612	0.069379015	98
IMBLearn Undersampling	0	0.999763765	**0.967518993**	0.983377125	56864
IMBLearn Undersampling	1	0.043995859	**0.867346939**	0.083743842	98
IMBLearn Oversampling	0	0.999727288	0.967009004	0.983095998	56864
IMBLearn Oversampling	1	0.042368555	0.846938776	0.080700049	98

Looking at the above table, I used the same test dataset with 56864 rows of valid transactions and 98 rows of fraudulent transactions.
This being a fraud detection model, we want to ensure that no fraud is undetected. It is ok if a valid transaction is marked as a fraud for further review, but we want to minimize the number of instances of fraud transactions that are undetected and are marked as valid.

So we want to reduce the instances of false negative, so we want to maximize the recall numbers here – so false negative are reduced.

Looking at the above table, **the IMBLearn Undersampled data results in the highest recall** for both the valid and fraudulent transactions.

Conclusions and Future Work

Implementing under sampling on the imbalanced dataset helped with the imbalance of the classes. We explored 2 different methods for dealing with imbalanced datasets Under sampling majority class and Oversampling minority class.

Additional enhancements are possible for this effort.

1. I could try using other methodologies such as Random Forest, GBM, neural networks to train the models.

Recommendations for the Clients

Based on my analysis, I would recommend the following:

1. The data set provided is very imbalanced. If possible, we should try to get few additional day's data that would increase the number of fraudulent transactions.
2. The following features seem to be most important to the outcomes.
3. It is recommended to use the imblearn package to undersample the data (if possible, with additional days of fraudulent data) to develop the model

Consulted Resources

DSC_CP_Final_Report_Suggested_Structure_V0_3_20_2020.txt
Displaying DSC_CP_Final_Report_Suggested_Structure_V0_3_20_2020.txt.